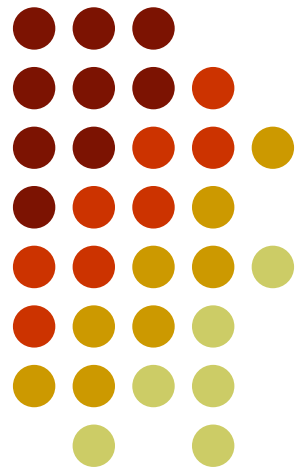
Web Server Architektur

Aysen Shibla
Daniel Förderer
Betreuer: Jürgen Vogel
Teleseminar Web Services (SS 04)
Universität Karlsruhe /
Universität Mannheim



Agenda

- Web Server
  - Definition
  - Arbeit
- Clustering
  - Cluster Architektur
  - Datentypen
  - Skalierbarkeit
  - Hardwaregestützte Cluster
  - Softwaregestützte Cluster
- Caching
  - Cooperative Caching
  - Replacement Algorithms
  - Evalutation

Web Server - Definition

Rechner, der Webseiten zum Abruf für Clients bereitstellt

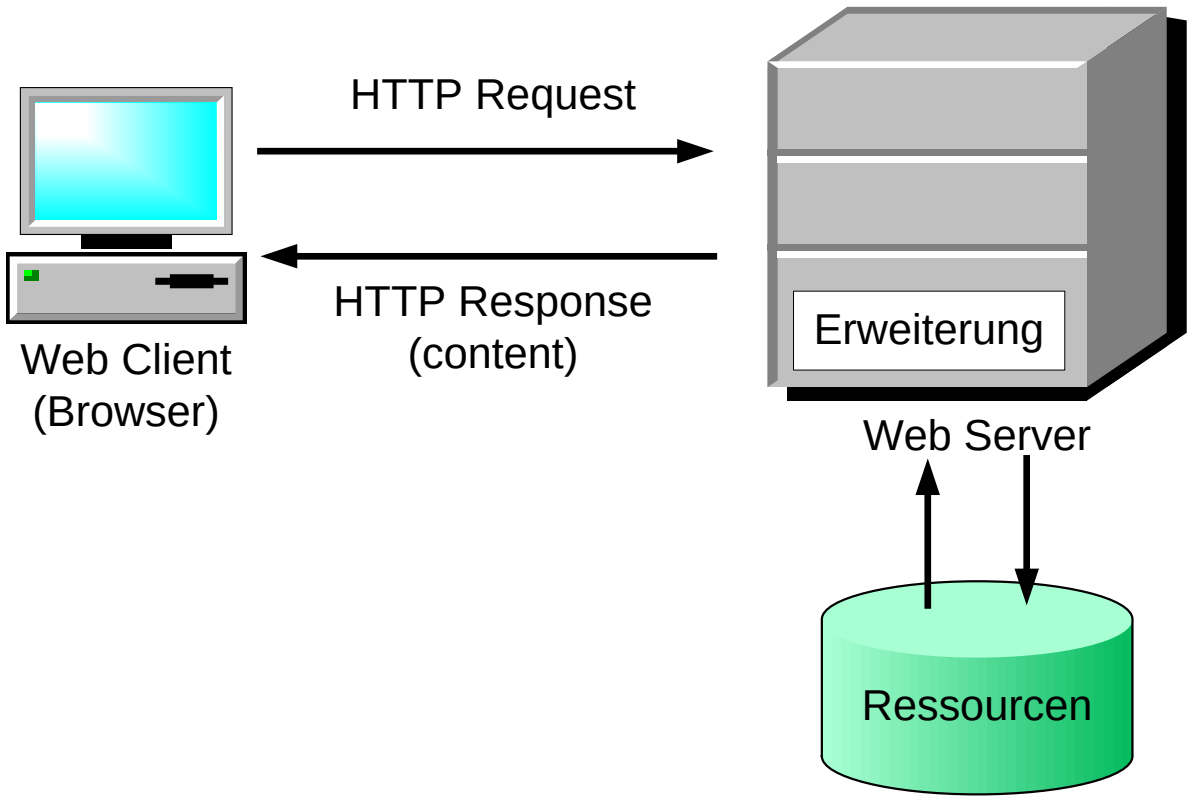
Web Server - zwei Begriffsverständnisse:

- Maschine selbst, auf dem entsprechende Software läuft
- Software, welche den Rechner befähigt (Beispiel: Apache)

Web Server - Verbindung

- Client und Server kommunizieren über das HyperText Transfer Protocol (HTTP)
  - Aufbau einer TCP Connection
  - Client fordert Service an
  - Server erledigt Arbeit und gibt Daten bzw. Fehlermeldung zurück
  - Beendigung der Connection (normalerweise)
- Aktuelle Version: HTTP/1.1
  - RFC 2616, Juni, 1999

Web Server - Aufgabe



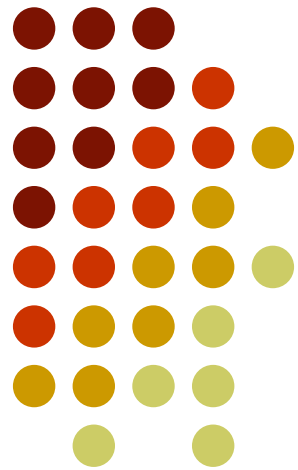
Web Server - Aufgabe

- Möglichkeiten der Web Server Erweiterung
  - Servlet
    - HTTP Requests werden von einem Java Programm (Servlet) mit speziellen Methoden beantwortet
    - Tomcat
    - Diverse Application Server
  - JSP
    - Java Code in HTML-Seiten eingebettet
    - Beim Aufruf einer JSP wird daraus ein Servlet generiert (compiliert)
    - Ausführung durch Servlet Engine

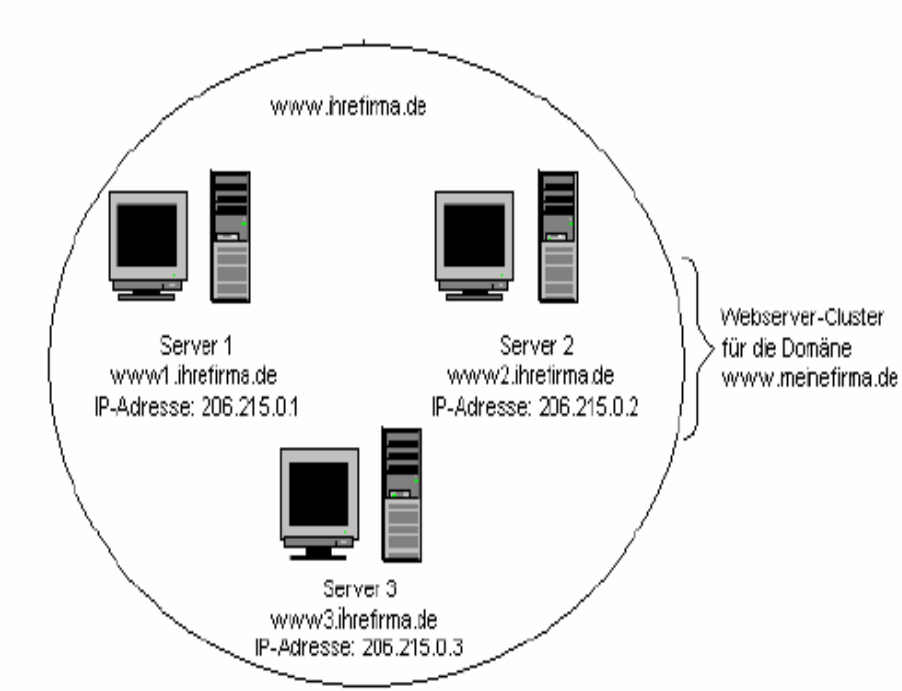
Web Server - Aufgabe

- Möglichkeiten der Erweiterung
  - PHP
    - In HTML eingebettetes PHP Programm wird beim Aufruf interpretiert
  - CGI
    - Perl Programme werden bei Aufruf gestartet und interpretiert
  - ASP
    - Microsoft Technologie, ähnlich wie Servlets in anderen Programmiersprachen

Clustering von Webservern



Clustering



„Unter Cluster versteht man eine lose gekoppelte Gruppe von Servern, die auf dieselben Daten zugreifen und dieselbe Gruppe von Clients bedienen kann.“

Ziel: Webdienste ohne Verluste

- Moderne Webdienste – 24 Stunden pro Tag , 7 tage pro Woche verfügbar
- Keine Ausfälle bei E-Commerce Diensten erwünscht
- Mehrere dynamische Seiten im WWW

Bedingungen:

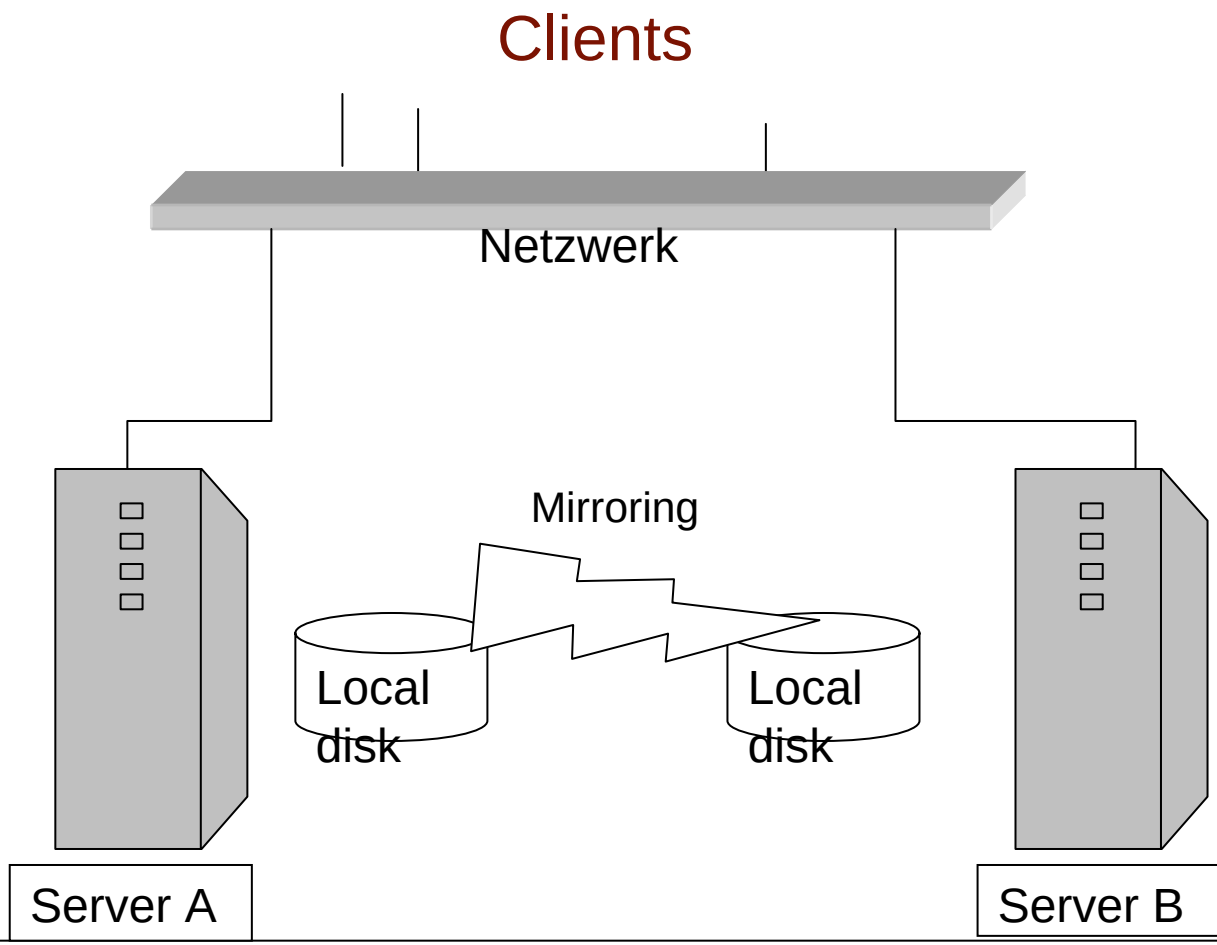
- Höhere Skalierbarkeit
- Verfügbarkeit
- Leichtere Verwaltung

des Clusters.

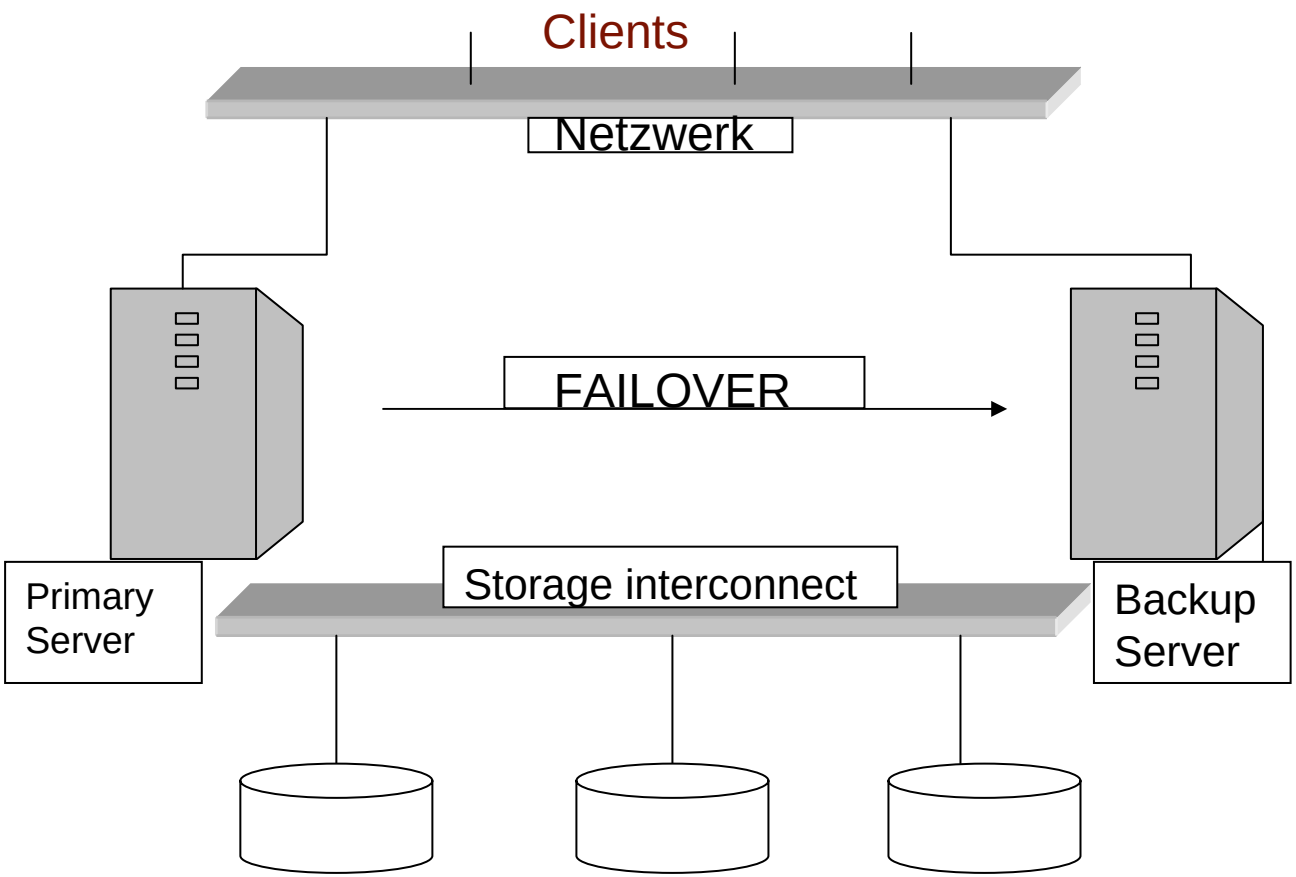
Cluster Architektur

- Share-Nothing & Shared-Storage
- Active-Active & Active-Standby

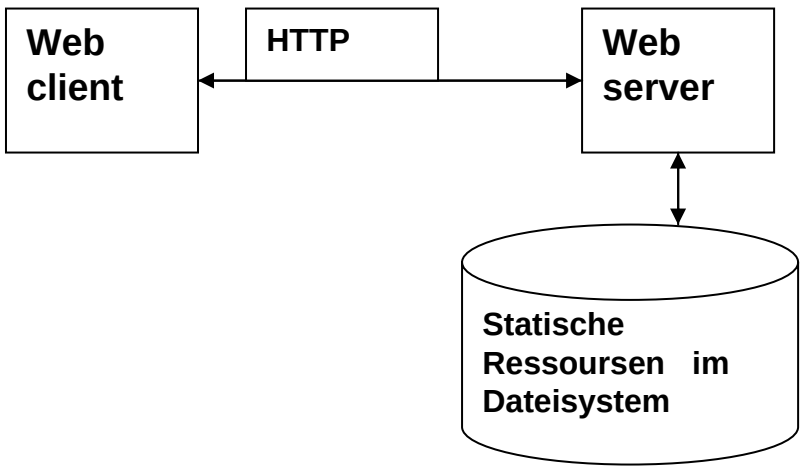
Share-nothing Clusters



Active- Standby

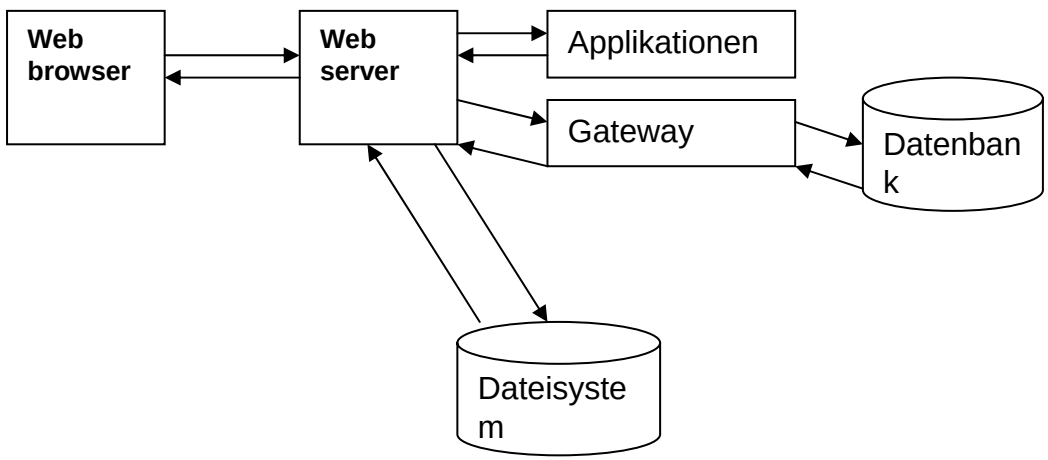


Statische Dateien



- Client fragt eine Ressource an
- Aufgrund des Bezeichners lokalisiert der Server die Ressource (z.B. im Dateisystem oder in der Datenbank)
- Server sendet die Ressource an Client zurück
- Client interpretiert die erhaltenen Dateien, z.B.
  - HTML darstellen
  - JavaScript interpretieren
  - Java Applet, ActiveX ausführen

Dynamische Dateien



- Client fragt eine Ressource an
- Aufgrund des Bezeichners erkennt Server,daß diese Ressource dynamisch erzeugt wird
- Ressource wird auf dem Server generiert
  - gegebenenfalls unter Zuhilfenahme der übergebenen Parameter
  - gegebenenfalls werden Seiteneffekte (z.B. anlegen einer
  - speichern eines Datensatz, steuern eines Geräts) erzeug
- Generierte Ressource wird an Client zurückgeschickt

Verfügbarkeit von Websites

- Jeder Zeit auf die Seite zugreifen können
- Trotz schwerer und ständig steigender Last verfügbar bleiben
- Keine Auswirkung eines Ausfalles

Häufige Ausfälle:

- Hardwareausfälle
- Softwareausfälle
- Serverausfälle

Failover

Möglichkeit einen Dienst beim Versagen des ausführenden Server von einem anderen Knoten übernehmen zu lassen

Skalierbarkeit

„ Die Fähigkeit eines Webservers die Verfügbarkeit, Zuverlässigkeit und Leistung einer Website beizubehalten, wenn die Anzahl der gleichzeitigen Webzugriffe, d.h. die Last für den Webserver, zunimmt.“

Häufige Engpässe:

- Prozessorkapazität
- Speicher
- Serverüberlastung

Skalierbarkeit

Leistung:

- Wie effizient eine Site auf Browser-Anforderungen antwortet hängt von der Anzahl der gleichzeitigen Verbindungen zum Webserver

Lastverwaltung:

- Verhindert Überlastung des Webservers

Lastverteilung:

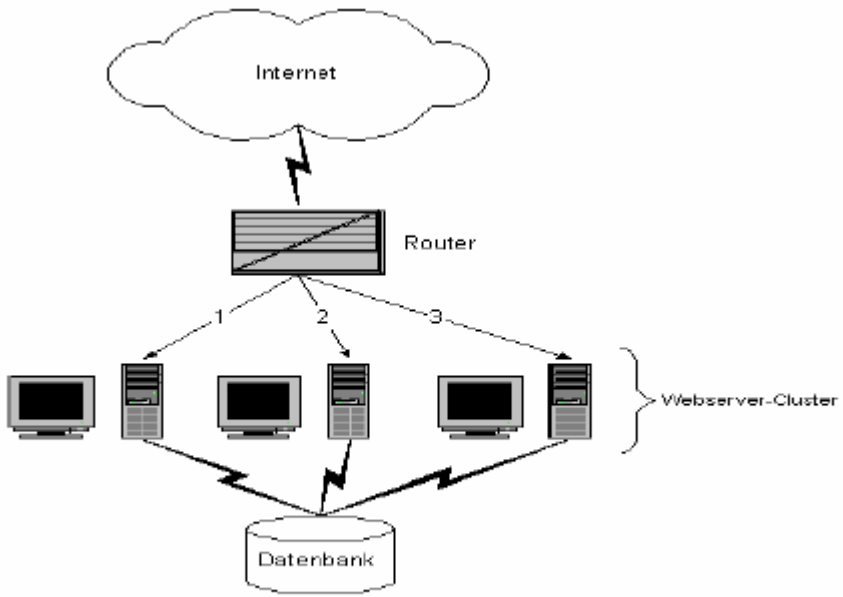
- eintreffende Anfragen auf allen Clusterknoten verteilen

Clustering von Webservern

„Technik, bei der zwei oder mehrere Server als Servercluster gruppiert werden.“

- **Hardwaregestütztes Clustering**
- **Softwaregestütztes Clustering**

Hardwaregestützte Clustering



Router:

Leitet eingehende HTTP Anforderungen an verfügbare Webserver im Cluster

- **Vorteile**:
  - kann Ausfälle erkennen
- **Nachteile:**
  - Single Point of Failure

Heartbeats

Softwaregestützte Clustering

Was ist DNS(Domain Name Server)?

- Verteilte Datenbank, deren Aufgabe ist, IP Adressen mit den Hostnamen zu verknüpfen

Hauptbestandteile von DNS:

- Zonen
- Domänen
- Namensserver

Softwaregestützte Clustering

Round - Robin DNS

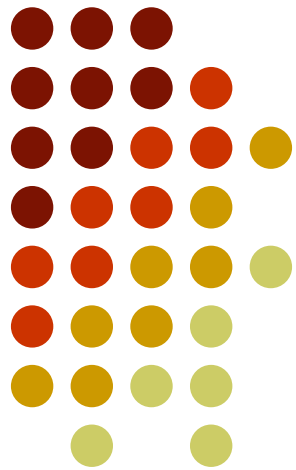
- Ein einziger Hostname zu mehreren IP Adressen verknüpfen
- Geeignet für Webserver, die nur eine statische Website zur Verfügung stellt.

Vorteile:

- einfach
- kostengünstig

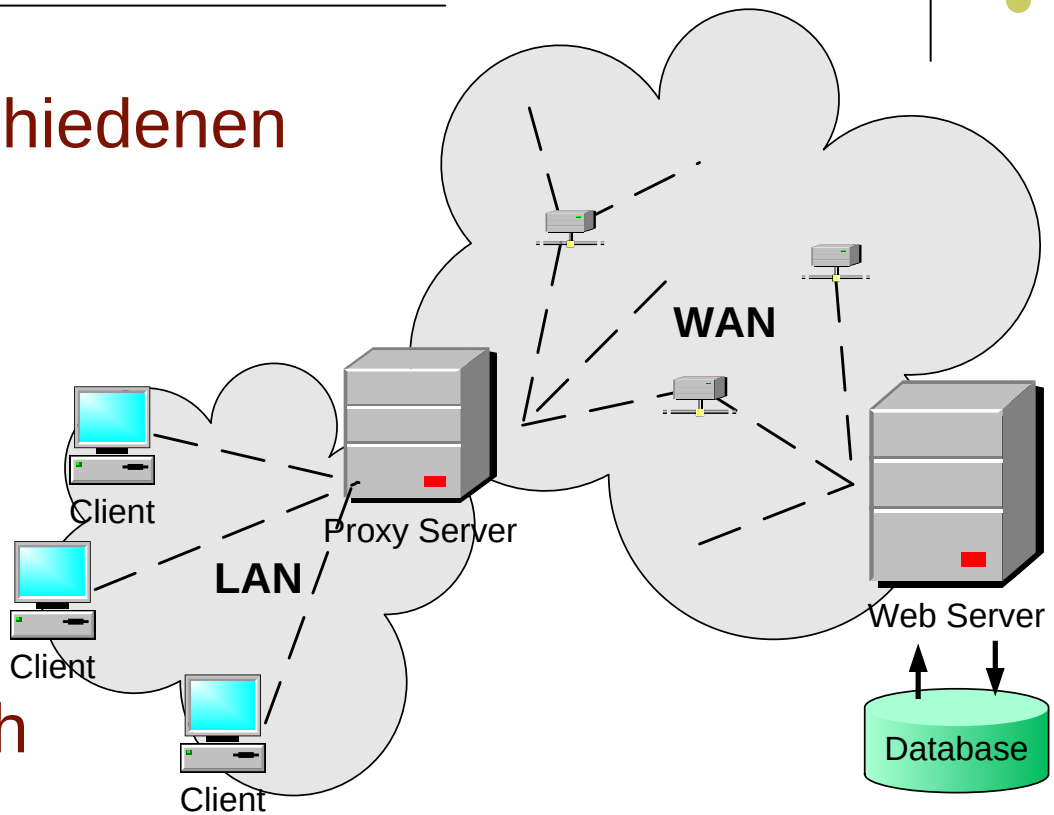
Nachteile: erkennt ausgefallene Server nicht

Caching



Caching

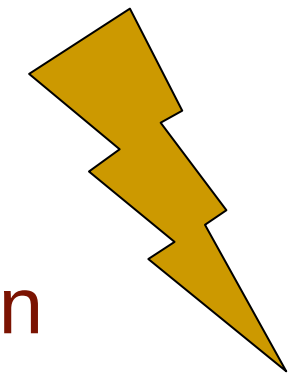
- Caching auf verschiedenen Ebenen möglich
  - Client
  - Proxy Server
  - Web Server
- Erhöht maßgeblich die Performance eines Web Servers



Caching – Web Objekte

- Web Objekte besitzen eine andere Granularität als herkömmliche Daten
- Cache Space ist knapp
  ⇨ Caching System muss abwägen:

Viele kleine Objekte im Cache speichern



Wenige große Objekte im Cache speichern

Web Objekte

- Statische Web Objekte
  - Keine Veränderungen über den Zeitverlauf
  - z.B. Bilder
- Dynamische Web Objekte
  - Besitzen nur eine begrenzte Lebenszeit, da sie aufgrund von Änderungen nicht mehr aktuell sind
  - Ständige Validation erforderlich um Cache Konsistenz aufrecht zu erhalten
  - z.B. Nachrichtenseite

Cache Replacement

- Mechanismus erforderlich, um den Inhalt des Caches zu verwalten
  - Wenig angefragte Objekte müssen zugunsten oft angefragter Objekte aus dem Cache gelöscht werden, um die „Hit Ratio“ zu erhöhen
  - Veraltete Web Objekte müssen entweder aktualisiert werden, oder aus dem Cache geworfen werden

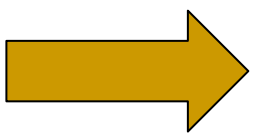
Caching im Cluster

- Lastverteilung auf verschiedene Server im Cluster verbessert noch nicht die Responsetime einer einzelnen Anfrage

- Erst das Caching spart Zeit, da langsame Remotezugriffe auf Ressource verringert werden

Cooperative Caching

- Cache eines jeden Servers für alle andere Server im Cluster verfügbar
- Zusammenlegung der gesamten Prozessorleistung und des Speicherplatzes im Cluster durch Cooperative Caching Algorithmus



Kernmechanismus des GOS

Quelle: „A scalable Cluster-based Web Server with Cooperative Aching Support“, by G.Chen, C.l.Wang and F.C.M.Lau, 2002

Cooperative Caching

- Vorteile
  - Netzwerkverbindung zwischen den im Cluster hängenden Server deutlich schneller als Datenbankanbindung
  - Cache Space wesentlich größer
  - I/O-Anfragen nehmen ab
  - „Hit Ratio“ verbessert sich
  - Client Wartezeit verringert sich

Global Object Space (GOS)

- Alle am Cluster beteiligten Server stellen einen Speicherblock für den gemeinsamen Cache des Clusters (GOS)
- Beinhaltet alle Web Objekte die sich im gemeinsamen Cache befinden

Hot Object Cache (HOC)

- Lokaler Cache eines jeden Cluster-Server
- Für alle im Cluster hängenden Server sowohl sichtbar, als auch zugänglich
- Sichtbarkeit und Zugang über speziell entworfenen Lookup-Mechanismus geregelt
- GOS setzt sich also aus allen HOCs zusammen

Home Node

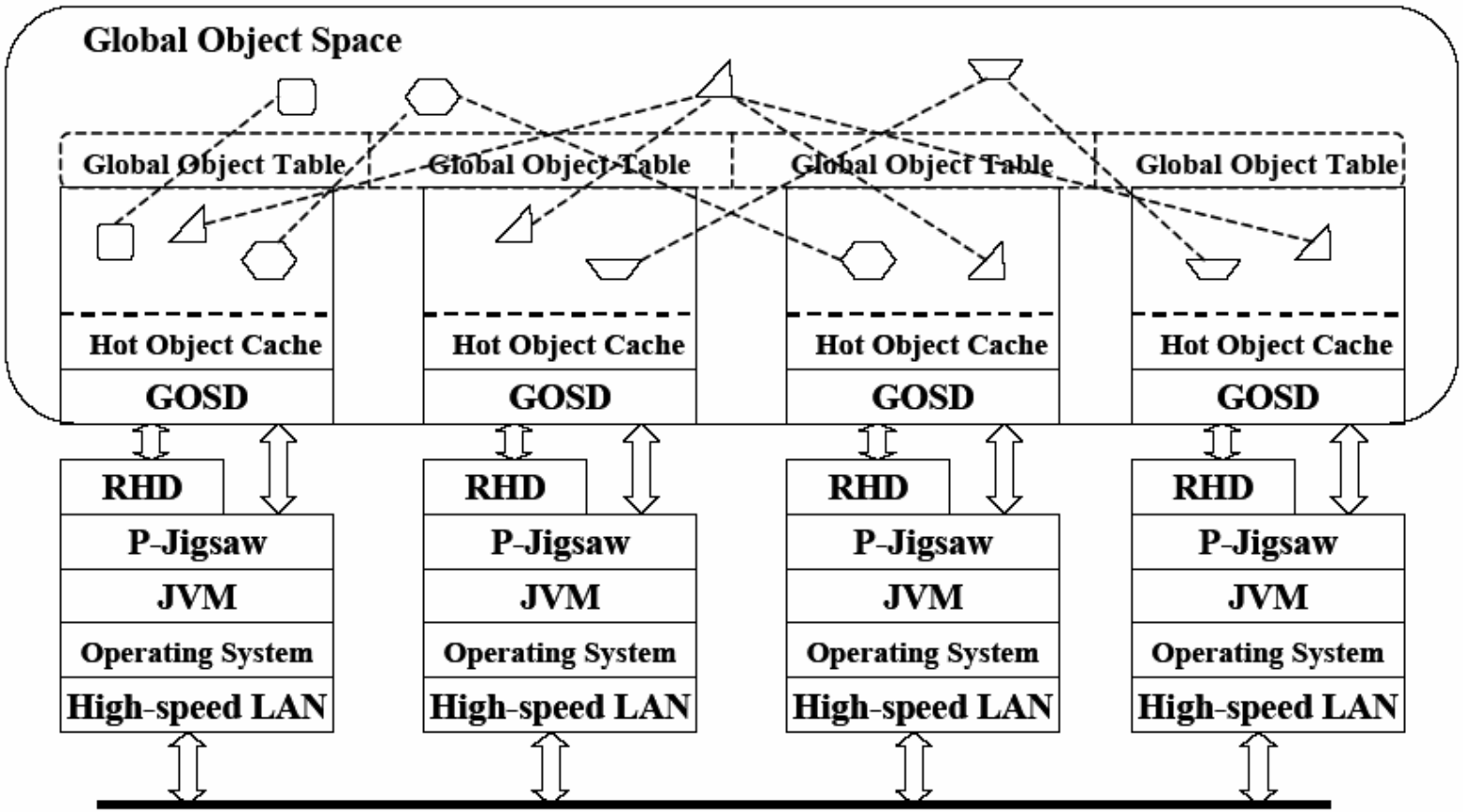
- Jedes Web Objekt hat einen eindeutigen Home Node im Cluster (Partition List)
- Hat als einziger Server Remote zugriff auf das Original des Web Objekts
- Jedes Web Objekt kann als „Hot Object“ in mehreren HOCs gleichzeitig sein
- Weiß in welchen HOCs das Objekt gerade präsent ist
- Verwaltet Zugriffsstatistik des Objektes

Bestimmung eines „Hot Objects“

Jedes Objekt besitzt zwei Parameter:

- AGAC (Approximated Global Access Counter)
  - Zugriffszähler des Web Objekts über den ganzen Cluster
  - Muss vom Home Node verwaltet werden
  - Alle LACs der jeweiligen HOS des Objekts fließen mit ein
- LAC (Local Access Counter)
  - Zugriffszähler des Objekts im jeweiligen HOS
  - Wird nach Update des AGAC immer wieder auf Null gesetzt
    - Periodisches Update
    - Threshold Überschreitungs-Update

HOC & GOS



Quelle: „A scalable Cluster-based Web Server with Cooperative Aching Support“, by G.Chen, C.l.Wang and F.C.M.Lau, 2002

GOSD and RHD

Jeder Server im Cluster operiert zwei Server Prozesse (Deamons)

- Global Object Space Service Deamin (GOSD)
  - Management des GOT (Global Object Table) und des LOT (Local Object Table)
  - Sicherung der Speicherstellen-Transparenz für alle Rechner im Cluster
  - Bearbeitung der Anfragen vom RHD und anderen GOSDs
- Request Handling Deamon (RHD)
  - Bearbeitung und Weiterleitung der über den TCP Port ankommenden HTTP Anfragen an den GOSD

LOT & GOT

GOSD managet zwei Table:

- LOT
  - lokale Zugriffsinformationen der Objekte im HOS
    - AGAC der Objekte (von Home Node erhalten)
    - LAC der fremden Objekte
    - Home Node IDs der fremden Objekte
- GOT
  - Abbilden von Objekt IDs (URLs) zu den Home Nodes
  - Verwaltet Globale Informationen als Home Node der zugehörigen Objekte
    - AGAC der eigenen Objekte
    - Server auf denen Objekte liegen

Cache Replacement Policies

- Least Recently Used (LRU)
  - Ältest benutztes Dokument im Cache wird verworfen
- Least Frequently Used (LFU)
  - Am wenigstens oft angefordertes Dokument wird verworfen
  - ⇨ Besser für Web Dokumente geeignet, da viele Requests von unterschiedlichen Clients eine hohe Popularität bedeuten

LFU Erweiterungen

- LFU-Aging
  - Idee: Popularitätsinformationen werden ohne Update über den Zeitverlauf immer unzuverlässiger
  - „Aging the AGAC“: AGAC wird am Ende eines festen Zeitintervalls halbiert
- Weighted-LFU
  - Web Objekte haben unterschiedlichen Speicherbedarf
  - Replacement Entscheidungen auf Basis der Größe und der Zugriffsfrequenz

Zwischenergebnis

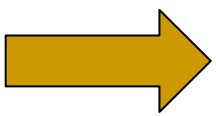
- Cache Replacement aufgrund niedriger „Hotness“ eines Objekts
  - Gilt soweit nur für statische Daten, die sich über den Zeitverlauf nicht ändern

Problem

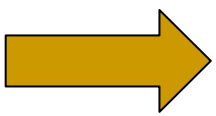
- Dynamische Daten werden auf Ressourcenseite ungültig (z.B. Nachrichten)
  - Invalidations-Mechanismus muss Sicherstellen, dass alle Kopien eines ungültigen Objekts ausgetauscht bzw. gelöscht werden

Refreshment dynamischer Objekte

- Content Misses
  - Objekt befindet sich nicht im Cache
- Freshness Misses (30-50% aller Cache Hits)
  - Objekt befand sich zwar im Cache, war aber veraltet
  - Validation des Objekts erforderlich



In beiden Fällen muss auf die Ressourcen zurückgegriffen werden



Solche Zugriffe sollen möglichst minimiert werden

Quelle: „Refreshment Policies for Web Content Caches“, by E.Cohen, H.Kaplan, 2001

Refreshment Policies

- Active Refreshment Policies
  - Validation erfolgt automatisch für lange abgelaufene Objekte im Cache (Predictive Policy)
- Passive Refreshment Policies
  - Validation erfolgt erst bei Freshness Miss

Refreshment Dynamischer Objekte

- Conditional Fetch
  - Cache sendet LMT (Last Modification Time) des Objekts zum Server
  - Server hat zwei Möglichkeiten
    - Validation der aktuellen Version
    - Senden komplett neuer Version
- Pure Validation Request
  - Cache fragt Server lediglich ob aktuelles Objekt noch gültig ist
  - Bei Nichtvalidität keine Versionserneuerung

Policies - Begriffe

- TTL (Time To Live)
  - Jedem Web Objekt wird Wert angehängt, welcher Lebenszeit angibt
- Renewal Credit
  - Zähler, der mit jeder neuen Validation aufgrund Ablauf der TTL heruntergezählt wird

Policies Approaches

- OPT(*i*)
  - Erfordert Wissen über zukünftige Objektnachfrage…
- RECENCY(*k*)
  - Renewal Credit wird mit jeder neuen Objekt Anfrage wieder auf k gesetzt (inkl. No-Cache-Requests)

Quelle: „Refreshment Policies for Web Content Caches“, by E.Cohen, H.Kaplan, 2001

Policies Approaches

- FREQ(j,m)
  - Mit jedem Freshness Hit wird m wieder auf Anfangswert gesetzt, falls er kleiner war
  - Renewal Credit m wird immer um j erhöht für jeden Freshness Miss (nicht bei Content Misses)
- TH-FREQ(th,m)
  - Mit jedem Freshness Hit wird der Renewal Credit wieder zurückgesetzt
  - Renewal Credit m wird heruntergezählt und Objekt neu geladen, wenn Rate unter Threshold fällt
  - Rate: Hit / TTL

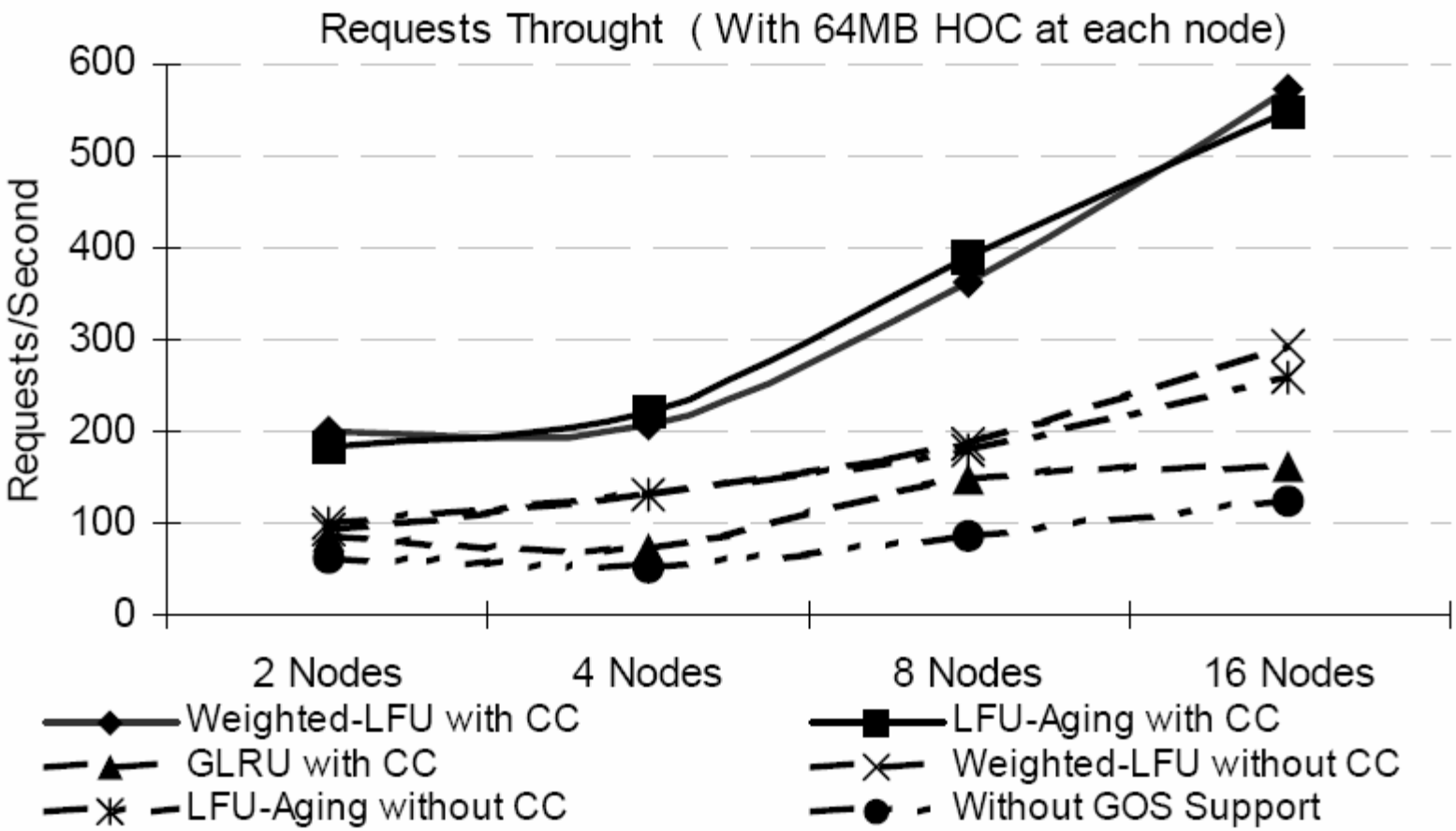
Ergebnisse einer Implementation

- Skalierung der Clustergröße
  - 64Mb pro Server
  - Von 2 auf 16 Server
- Ergebnisse Request Throughput (RT):
  - Ohne GOS Support: Erhöhung um Faktor 2.02
  - Mit CC und Weighted-LFU: 2.89

[Nachträglicher Vergleich:
  - CC mit LRU: 1.71
  - CC mit Weighted-LFU: 2.89]

Quelle: „A scalable Cluster-based Web Server with Cooperative Aching Support“, by G.Chen, C.l.Wang and F.C.M.Lau, 2002

Effects Of Scaling Cluster Size



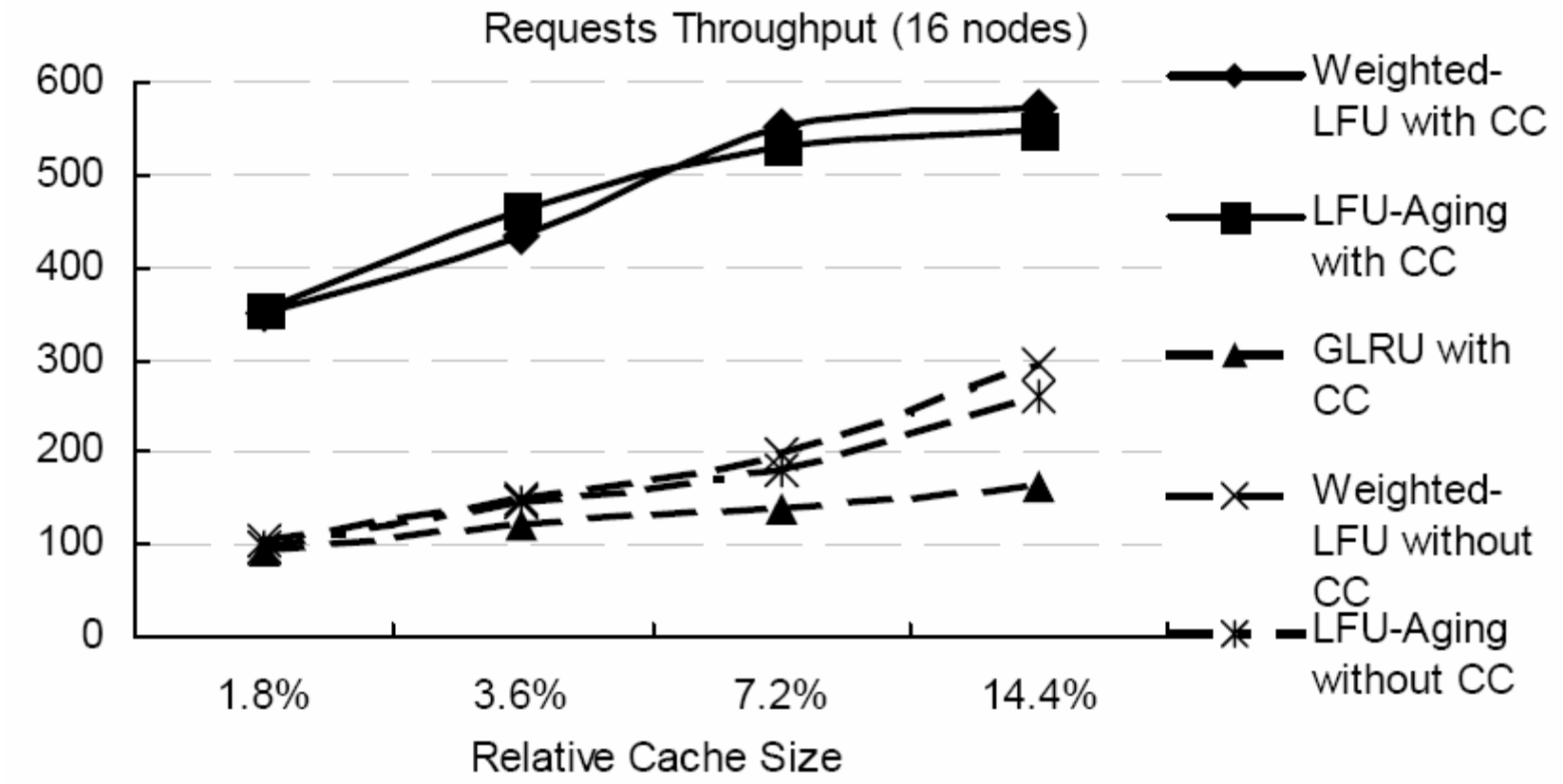
Quelle: „A scalable Cluster-based Web Server with Cooperative Aching Support“, by G.Chen, C.I.Wang and F.C.M.Lau, 2002

Ergebnisse einer Implementation

- Skalierung der Cachegröße
  - Von 8Mb auf 64Mb
  - 16 Server
  - Relative Cache Size (RCS): 1.8% -14.4%
  - CC mit Weighted-LFU
- Ergebnisse des (RT):
  - RCS = 1.8% ⇨ 73% Global Hit Rate (GHR)
  - RCS = 14.4% ⇨ 87% GHR
  - Maximaler Wert ohne CC: 55%

[GHR von Weighted-LFU als bei LFU-Aging
  - Vor Allem bei kleiner RCS]

Effects Of Scaling Cache Size



Quelle: „A scalable Cluster-based Web Server with Cooperative Aching Support“, by G.Chen, C.I.Wang and F.C.M.Lau, 2002

Server mit CC-Unterstützung

- ***Distributed Cooperative* Apache (DC-Apache)**
  - „Distributed Cooperative Apache Web Server“, by Li, Moon, 2002
- **Whizz Technology‘s WhizzBee Web Server**
  - http://www.whizztech.com

- **Unterstützende Technologien:**
  - **Location Aware Request Distribution (LARD)**
    - Dispatcher Mechanismus des Clusters
    - „Locality-aware Request Distribution in Cluster-based Network Servers“, by Pai, Aron, 1998
  - **Duplicate Copy First Replacement (DFR)**
    - Schutz vor der Löschung eines Singlets im Cache
    - „Efficient Cooperative Caching for File Systems in Cluster-based Web Servers“, by Ahn, Park S., Park D., 2000

Zusammenfassung

Responsetime abhängig von:

- Rechenleistung
  - Verbesserung durch Clustering von Servern (Load-Balancing)
- Hit Ratio
  - Verbesserung durch intelligenten Cooperative Cache Algorithmus
  - Verbesserung durch intelligente Refreshment Policy

Further Readings

- Cooperative Caching
  - „A scalable Cluster-based Web Server with Cooperative Aching Support“, by G.Chen, C.I.Wang and F.C.M.Lau, 2002
- Refreshment Policies
  - „Refreshment Policies for Web Content Caches“, by E.Cohen, H.Kaplan, 2001

ENDE

Vielen Dank für eure Aufmerksamkeit